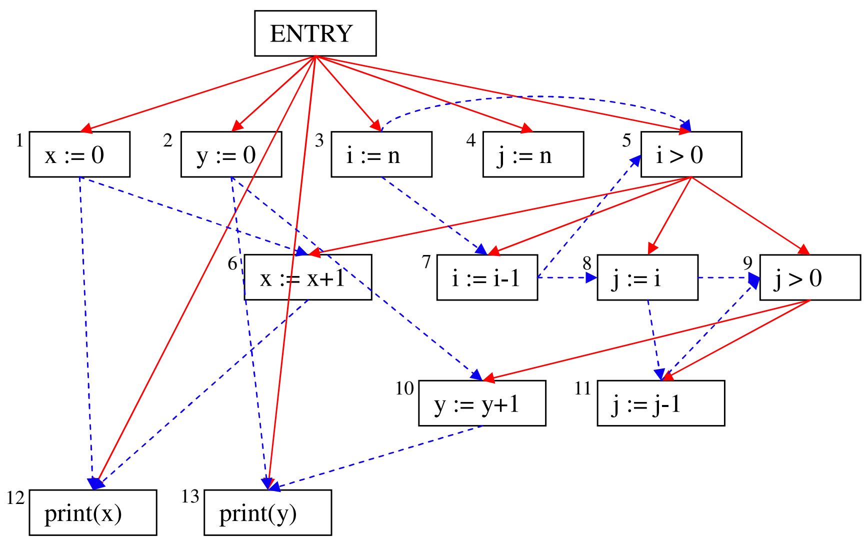
Problem Sheet 7: Program Slicing and Abstract Interpretation Sample Solutions

Chris Poskitt*
ETH Zürich

Starred exercises (∗) are more challenging than the others.

1 Program Slicing

i. Here is the program dependence graph for the program fragment (blue arrows are from the use-definition analysis; red arrows indicate control dependencies):



ii. For slicing criterion `print(x)`, i.e. block 12, we get:

```
x := 0;
i := n;
while i > 0 do
     x := x + 1;
     i := i - 1;
end
print(x);
```

*These solutions are adapted from previous iterations of the course when Stephan van Staden was the teaching assistant.

For slicing criterion `print(y)`, i.e. block 13, we get:

```
y := 0;
i := n;
while i > 0 do
     i := i - 1;
     j := i;
     while j > 0 do
          y := y + 1;
          j := j - 1;
     end
end
print(y);
```

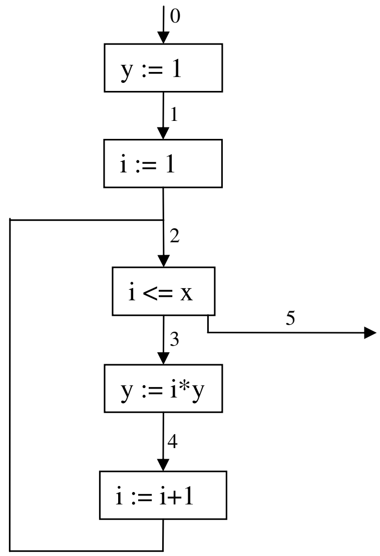
2 Abstract Interpretation

i. We begin by mapping every variable to $\bot$ (except for $\mathtt{x}, \mathtt{y}$ in A_1, which are respectively mapped to $\mathtt{+}, \top$ by assumption). Then, we iteratively update the (abstract) values of variables by applying the system of equations.

Abstract States	Iterations $\longrightarrow$													Final Values
$A_1(\mathtt{x})$	+													+
$A_1(\mathtt{y})$	$\top$													$\top$
$A_2(\mathtt{x})$	$\bot$	+				$\top$				$\top$				$\top$
$A_2(\mathtt{y})$	$\bot$	+				+				$\top$				$\top$
$A_3(\mathtt{x})$	$\bot$		+				$\top$				$\top$			$\top$
$A_3(\mathtt{y})$	$\bot$		+				+				$\top$			$\top$
$A_4(\mathtt{x})$	$\bot$			+				$\top$				$\top$		$\top$
$A_4(\mathtt{y})$	$\bot$			+				$\top$				$\top$		$\top$
$A_5(\mathtt{x})$	$\bot$				$\bot$				0				0	0
$A_5(\mathtt{y})$	$\bot$				+				+				$\top$	$\top$

ii. The analysis is not very precise: it cannot prove that `y` is positive when the program fragment completes (i.e. at A_5).

iii. (a) If we compute the factorial using a program that does not utilise the subtraction operator, then the result of the analysis becomes more precise:



$A_0 = [x \mapsto +, y \mapsto \top, i \mapsto \top]$
$A_1 = A_0[y \mapsto +]$
$A_2 = A_1[i \mapsto +] \sqcup A_4[i \mapsto A_4(i) \oplus +]$
$A_3 = A_2$
$A_4 = A_3[y \mapsto A_3(i) \otimes A_3(y)]$
$A_5 = A_2$

Abstract States	Final Values
$A_0(\mathtt{x})$	+
$A_0(\mathtt{y})$	$\top$
$A_0(\mathtt{i})$	$\top$
$A_1(\mathtt{x})$	+
$A_1(\mathtt{y})$	+
$A_1(\mathtt{i})$	$\top$
$A_2(\mathtt{x})$	+
$A_2(\mathtt{y})$	+
$A_2(\mathtt{i})$	+
$A_3(\mathtt{x})$	+
$A_3(\mathtt{y})$	+
$A_3(\mathtt{i})$	+
$A_4(\mathtt{x})$	+
$A_4(\mathtt{y})$	+
$A_4(\mathtt{i})$	+
$A_5(\mathtt{x})$	+
$A_5(\mathtt{y})$	+
$A_5(\mathtt{i})$	+

(b) (∗) Perhaps changing the program for the analysis to work more precisely is not the best approach—let's try to improve the analysis! We'll try a so-called *relational analysis* with domain $\mathfrak{P}(\{\texttt{-}, \texttt{0}, \texttt{+}\} \times \{\texttt{-}, \texttt{0}, \texttt{+}\})$ to represent program states $(\mathtt{x}, \mathtt{y})$. A relational analysis is more precise because the domain can express dependencies, or relationships, between `x` and `y`.

We use the original version of the program fragment, but the new system of equations below:

$$
\begin{aligned}
A_1 &= \{(+,-), (+,0), (+,+)\} \\
A_2 &= \{(x,+) \mid (x,y) \in A_1\} \cup \{(x,y') \mid (x',y') \in A_4 \text{ and } x \in x' \ominus +\} \\
A_3 &= A_2 \cap \{(x,y) \mid x \in \{-,+\} \text{ and } y \in \{-,0,+\}\} \\
A_4 &= \{(x',y) \mid (x',y') \in A_3 \text{ and } y \in x' \otimes y'\} \\
A_5 &= A_2 \cap \{(0,y) \mid y \in \{-,0,+\}\}
\end{aligned}
$$

and obtain a more precise analysis allowing us to deduce that `y` will be positive after execution finishes:

	Iterations							Answer
A_1	{(+,-), (+,0), (+,+)}						...	{(+,-),(+,0), (+,+)}
A_2	Ø	{(+,+)}			{(+,+),(0,+), (-,+)}		...	{(+,+),(-,+), (0,+),(-,-)}
A_3	Ø		{(+,+)}			{(+,+), (-,+)}	...	{(+,+),(-,+), (-,-)}
A_4	Ø			{(+,+)}			...	{(+,+),(-,-), (-,+)}
A_5	Ø						...	{(0,+)}